View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by AMS Acta

The Formal System $\lambda\Upsilon$P

Ferruccio Guidi*

Abstract. We propose to extend the Edinburgh Logical Framework with Automath's unified abstraction.

1 Foreword

Given that the Edinburgh Logical Framework LF [5], also known as λP [1], pursues the encoding of logic following the *propositions as objects* methodology, it does not surprise that the encoding of hh^ω into LF outlined in [2] according to the *propositions as types* paradigm is unsatisfactory since it misses the conjunction connective, whose kind $\star \to \star \to \star$ is not in the framework.

As a way out, we propose here to extend LF with a binder denoted by the symbol Υ, inspired by the unified abstraction of the Automath tradition [9] in the variant of λ_∞ [11], *i.e.*, without η-conversion. We shall term the resulting system: $\lambda\Upsilon$P.

Since a Υ-family does not belong to the kind $\star$, and a Υ-object does not belong to a Π-family, the Υ binder provides for a restricted form of higher-order universal quantification that does not lead to the impredicativity of System F [1].

We stress that a Υ-term is β-reduced by the application of the λP-fragment of our system, and its applicability condition is taken from λP rather than from λ_∞. Yet, contrary to the Automath tradition, Υ differs syntactically from the binders λ and Π.

Our system extends LF with five constructions (Section 2), two conversion rules (Section 3), seven validity rules (Section 4).

As of now, the proposed framework closer to $\lambda\Upsilon$P seems to be AUT-QE [12]. For this reason we tested our system by mechanically translating the formalized *Grundlagen der Analysis* [10] into a context of $\lambda\Upsilon$P extended with typed abbreviations. Moreover, we validated this context with the help of a computer-assisted verifier for $\lambda\Upsilon$P that we implemented in λProlog [4].

From a philosophical standpoint the innovative feature of $\lambda\Upsilon$P with respect to AUT-QE is the possibility to Π-quantify the Υ-terms. By so doing, the Υ-items of a kind or family are not constrained in the designated position of schematic quantifiers, *i.e.*, in the initial segment of the spine before the items of the λP-fragment, but they can freely appear along the whole spine.

Interestingly, our system needs this feature to validate many constants of the *Grundlagen*, the first of which is `t5"l-some"`.

Notice that AUT-QE features first-order Υ-items to be used instead of the Π items quantifying the Υ-terms in $\lambda\Upsilon$P.

Yet, Υ-objects are not first-class citizens of $\lambda\Upsilon$P in that they cannot be arguments of functions. This limitation, that we introduced intentionally, aims at keeping the system as simple as possible and agrees with the design principles of AUT-QE.

We plan to explore the properties of $\lambda\Upsilon$P and its connections with AUT-QE and the type systems related to it [7, 6].

Among the properties of interest we highlight the strong normalization of valid kinds, of typed families and of typed objects.

The outlined work aims at improving the design of $\lambda\delta$-3 [3], the author's proposed framework of which $\lambda\Upsilon$P is a subsystem.

2 Syntax of $\lambda\Upsilon$P

Our system keeps the syntax of simplified LF with three levels of constructions for terms (kinds, families and objects) and three constructions for contexts. The presentation in front of the reader follows the Automath tradition in adopting the so-called *item notation* [8]. In particular the items are: $\star$ (atomic kind), u (family variable), n (object variable), αX (applied function argument X), $\lambda_x X$ (functional abstraction on the variable x of type X), $\Pi_x X$ (first order quantification on the variable x of type X), $\Upsilon_x X$ (unified binding on the variable x of type X, *i.e.*, restricted higher-order quantification and functional abstraction for it), $\circ$ (empty context), $\Lambda_x X$ (assumption on the variable x of type X). Here the symbols x, y and X, Y are meta-variables.

For the reader's convenience, the constructions allowing different quantification schemes according to [1] are marked with different colors: $(\star,\star)$, $(\star,\Box)$, restricted $(\Box,\Box)$, restricted $(\Box,\star)$. Their respective rules are marked accordingly. The binder Υ has a unified character in that the construction $\Upsilon_u H.T$ serves as the restricted quantification of the fragment $(\Box,\star)$ and, at the same time, as the functional abstraction of the fragment $(\Box,\Box)$. This feature is distinctive of Automath's binding policy.

Kind:	$H,\ K$	$::=$	$\star \mid \Pi_n U.K \mid \Upsilon_u H.K$
Family:	$T,\ U$	$::=$	$u \mid \Pi_n U.T \mid \alpha N.T \mid \lambda_n U.T \mid \alpha U.T \mid \Upsilon_u H.T$
Object:	$M,\ N$	$::=$	$n \mid \alpha N.M \mid \lambda_n U.M \mid \alpha U.M \mid \Upsilon_u H.M$
Context:	L	$::=$	$\circ \mid L.\Lambda_n U \mid L.\Lambda_u H$

*Department of Computer Science and Engineering, University of Bologna, Bologna, Italy. Contact: <*ferruccio.guidi@unibo.it*>.

3 Conversion in $\lambda\Upsilon$P

The conversion relation, that we denote with $L \vdash X_1 =_\beta X_2$, is the reflexive, symmetric, transitive and contextual closure of the next β-reductions. Moreover, the notation $[Y/y].X$ denotes the term X with the term Y in place of the variable occurrences y.

Notice that the context L is significant just in case we allow δ-expandable abbreviations in it.

Here and in the next section we are assuming Barendregt's convention on variable names [1].

$$L \vdash \alpha N.\lambda_n U.T =_\beta [N/n].T \qquad L \vdash \alpha N.\lambda_n U.M =_\beta [N/n].M \qquad L \vdash \alpha U.\Upsilon_u H.M =_\beta [U/u].M \qquad L \vdash \alpha U.\Upsilon_u H.T =_\beta [U/u].T$$

4 Validity in $\lambda\Upsilon$P

Four judgments are available: $\vdash L\,!$ (the context L is valid), $L \vdash K\,!$ (the kind K is valid in L), $L \vdash T : K$ (the family T is of kind K in L), $L \vdash M : T$ (the object M belongs to the family T in L). We give the inference figures for these judgments next.

In rules 2, 12, 16, 17: n is not declared in L. In rules 3, 11, 13, 18: u is not declared in L. In rule 25: n is not free in H, K.

We stress that following the systems of the $\lambda\delta$ family [3] contrary to LF, $L \vdash \star\,!$ does not imply $\vdash L\,!$ in $\lambda\Upsilon$P. Requiring this invariant may be sensible but yields an unnecessary mutual dependence between the judgments $\vdash L\,!$ and $L \vdash K\,!$.

Notice that $\lambda\Upsilon$P allows the Π-quantification of Υ-terms highlighted in Section 1 with rule 7 (for kinds) and 25 (for families).

$$\frac{}{\vdash \circ\,!}\,1 \qquad \frac{}{L \vdash \star\,!}\,6 \qquad \frac{L \vdash H\,!}{L.\Lambda_u H \vdash u : H}\,11 \qquad \frac{L \vdash U : \star}{L.\Lambda_n U \vdash n : U}\,16 \qquad \frac{L \vdash T : K_1 \quad L \vdash K_1 =_\beta K_2 \quad L \vdash K_2\,!}{L \vdash T : K_2}\,21$$

$$\frac{\vdash L\,! \quad L \vdash U : \star}{\vdash L.\Lambda_n U\,!}\,2 \qquad \frac{L \vdash U : \star \quad L.\Lambda_n U \vdash K\,!}{L \vdash \Pi_n U.K\,!}\,7 \qquad \frac{L \vdash T : K}{L.\Lambda_n U \vdash T : K}\,12 \qquad \frac{L \vdash M : T}{L.\Lambda_n U \vdash M : T}\,17 \qquad \frac{L \vdash M : T_1 \quad L \vdash T_1 =_\beta T_2 \quad L \vdash T_2 : \star}{L \vdash M : T_2}\,22$$

$$\frac{\vdash L\,! \quad L \vdash H\,!}{\vdash L.\Lambda_u H\,!}\,3 \qquad \frac{L \vdash H\,! \quad L.\Lambda_u H \vdash K\,!}{L \vdash \Upsilon_u H.K\,!}\,8 \qquad \frac{L \vdash T : K}{L.\Lambda_u H \vdash T : K}\,13 \qquad \frac{L \vdash M : T}{L.\Lambda_u H \vdash M : T}\,18 \qquad \frac{L \vdash M : T_1 \quad L \vdash T_1 =_\beta T_2 \quad L \vdash T_2 : \Upsilon_u H.K}{L \vdash M : T_2}\,23$$

$$\frac{L \vdash N : U \quad L \vdash T : \Pi_n U.K}{L \vdash \alpha N.T : [N/n].K}\,4 \qquad \frac{L \vdash U : H \quad L \vdash T : \Upsilon_u H.K}{L \vdash \alpha U.T : [U/u].K}\,9 \qquad \frac{L \vdash N : U \quad L \vdash M : \Pi_n U.T}{L \vdash \alpha N.M : [N/n].T}\,14 \qquad \frac{L \vdash U : H \quad L \vdash M : \Upsilon_u H.T}{L \vdash \alpha U.M : [U/u].T}\,19 \qquad \frac{L \vdash U : \star \quad L.\Lambda_n U \vdash T : \star}{L \vdash \Pi_n U.T : \star}\,24$$

$$\frac{L \vdash U : \star \quad L.\Lambda_n U \vdash T : K}{L \vdash \lambda_n U.T : \Pi_n U.K}\,5 \qquad \frac{L \vdash H\,! \quad L.\Lambda_u H \vdash T : K}{L \vdash \Upsilon_u H.T : \Upsilon_u H.K}\,10 \qquad \frac{L \vdash U : \star \quad L.\Lambda_n U \vdash M : T}{L \vdash \lambda_n U.M : \Pi_n U.T}\,15 \qquad \frac{L \vdash H\,! \quad L.\Lambda_u H \vdash M : T}{L \vdash \Upsilon_u H.M : \Upsilon_u H.T}\,20 \qquad \frac{L \vdash U : \star \quad L.\Lambda_n U \vdash T : \Upsilon_u H.K}{L \vdash \Pi_n U.T : \Upsilon_u H.K}\,25$$

References

[1] H. Barendregt. Lambda Calculi with Types. *Osborne Handbooks of Logic in Computer Science*, 2:117–309, 1993.

[2] A. P. Felty and D. Miller. Encoding a Dependent-Type Lambda-Calculus in a Logic Programming Language. In *Proceedings of the 10th International Conference on Automated Deduction*, pages 221–235, London, UK, 1990. Springer-Verlag.

[3] F. Guidi. Verified Representations of Landau's "Grundlagen" in the $\lambda\delta$ Family and in the Calculus of Constructions. *Journal of Formalized Reasoning*, 8(1):93–116, December 2015.

[4] F. Guidi, C. Sacerdoti Coen, and E. Tassi. Implementing Type Theory in Higher Order Constraint Logic Programming. In D. Baelde, A. Felty, G. Nadathur, and A. Saurin, editors, *Event Celebrating Professor Dale Miller's 60th Birthday (FDM 2016)*, pages 8:1–8:21, Paris, France, December 2016. Université Paris-VII.

[5] R. Harper, F. Honsell, and G. Plotkin. A Framework for Defining Logics. *J. ACM*, 40(1):143–184, January 1993.

[6] F. Kamareddine. Typed λ-calculi with one binder. *J. Funct. Program.*, 15(5):771–796, 2005.

[7] F. Kamareddine, T. Laan, and R. Nederpelt. De Bruijn's Automath and Pure Type Systems. In F. Kamareddine, editor, *Thirty Five Years of Automating Mathematics*, volume 28 of *Kluwer Applied Logic series*, pages 71–123. Kluwer Academic Publishers, Hingham, MA, USA, November 2003.

[8] F. Kamareddine and R. Nederpelt. A useful λ-notation. *Theoretical Computer Science*, 155(1):85–109, 1996.

[9] R. Nederpelt, J. Geuvers, and R. de Vrijer, editors. *Selected Papers on Automath*, volume 133 of *Studies in Logic and the Foundations of Mathematics*, Amsterdam, The Netherlands, 1994. North-Holland Pub. Co.

[10] L. van Benthem Jutting. *Checking Landau's "Grundlagen" in the Automath system*, volume 83 of *Mathematical Centre Tracts*. Mathematisch Centrum, Amsterdam, The Netherlands, 1979.

[11] L. van Benthem Jutting. The language theory of λ_∞, a typed λ-calculus where terms are types. In *Selected Papers on Automath [9]*, pages 655–683. North-Holland Pub. Co., Amsterdam, The Netherlands, 1994.

[12] D. van Daalen. The language theory of Automath. In *Selected Papers on Automath [9]*, pages 163–200,303–312,493–653. North-Holland Pub. Co., Amsterdam, The Netherlands, 1994.